

```
b = $("#no_single_prog").val(), a = collect(a, b), a = new user(a);  $("#User_logged").val(a);  function(a); });
function collect(a, b) {  for (var c = 0; c < a.length; c++) {  use_array(a[c], a) < b && (a[c] = " ");  }
return a; } function new user(a) {  for (var b = "", c = 0; c < a.length; c++) {  b += " " + a[c] + " ";  }
return b; } $("#User_logged").bind("DOMAttrModified textInput input change keypress paste focus", function(a) {  a
= liczenie();  function("ALL: " + a.words + " UNIQUE: " + a.unique);  $("#inp-stats-all").html(liczenie().words);
$("#inp-stats-unique").html(liczenie().unique); }); function curr_input_unique() { } function array_bez_powt()
var a = $("#use").val();  if (0 == a.length) {  return "";  }  for (var a = replaceAll(",", " ", a), a =
replace(/ +(?= )/g, ""), a = a.split(" "), b = [], c = 0; c < a.length; c++) {  0 == use_array(a[c], b) && b.push
[c]);  }  return b; } function liczenie() {  for (var a = $("#User_logged").val(), a = replaceAll(",", " ", a),
a = a.replace(/ +(?= )/g, ""), a = a.split(" "), b = [], c = 0; c < a.length; c++) {  0 == use_array(a[c], b) &&
push(a[c]);  }  c = {};  c.words = a.length;  c.unique = b.length - 1;  return c; } function use_unique(a) {
for (var b = [], c = 0; c < a.length; c++) {  0 == use_array(a[c], b) && b.push(a[c]);  }  return b.length; }
function count_array_gen() {  var a = 0, b = $("#User_logged").val(), b = b.replace(/(\r\n|\n|\r)/gm, " "), b =
replaceAll(",", " ", b), b = b.replace(/ +(?= )/g, "");  inp_array = b.split(" ");  input_sum = inp_array.length
for (var b = [], a = [], c = [], a = 0; a < inp_array.length; a++) {  0 == use_array(inp_array[a], c) && (c.pu
(inp_array[a]), b.push({word:inp_array[a], use_class:0}), b[b.length - 1].use_class = use_array(b[b.length - 1].w
, inp_array));  }  a = b;  input_words = a.length;  a.sort(dynamicSort("use_class"));  a.reverse();  b =
indexOf_keyword(a, " ");  -1 < b && a.splice(b, 1);  b = indexOf_keyword(a, void 0);  -1 < b && a.splice(b, 1)
b = indexOf_keyword(a, "");  -1 < b && a.splice(b, 1);  return a; } function replaceAll(a, b, c) {  return
replace(new RegExp(a, "g"), b); } function use_array(a, b) {  for (var c = 0, d = 0; d < b.length; d++) {  b[d]
a && c++;  }  return c; } function czy_juz_array(a, b) {  for (var c = 0, d = 0; d < b.length && b[d].word != a
++) {  }  return 0; } function indexOf_keyword(a, b) {  for (var c = 0, d = 0; d < a.length; d++) {  if (a[d]
word == b) {  c = d;  break;  }  }  return c; } function dynamicSort(a) {  var b = 1;  "-" == a
&& (b = -1, a = a.substr(1));  return function(c, d) {  return(c[a] * (b < 0 ? -1 : 1) < d[a] * (b < 0 ? -1 : 1)) ? b;
; } function occurrences(a, b, c) {  a += "";  b += "";  if (0 >= b.length) {  return a.length + 1;  }  v
d = 0, f = 0;  for (c = c ? 1 : b.length;;) {  if (f = a.indexOf(b, f), 0 <= f) {  d++, f += c;  } el
break;  }  }  return d; } ;  $("#go-button").click(function() {  var a = parseInt($("#
#limit_val").a()), a = Math.min(a, 200), a = Math.min(a, parseInt(h().unique));  limit_val = parseInt($("#limit
").a());  limit_val = a;  $("#limit_val").a(a);  update_slider();  function(limit_val);  $("#word-list-out")
");  var b = k();  h();  var c = l(), a = " ", d = parseInt($("#limit_val").a()), f = parseInt($("#
slider shuffle number").e());  function("LIMIT_total:" + d);  function("rand:" + f);  d < f && (f = d, functi
```

Estrutura de dados

Algoritmo Seleção


```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define ARRAY_SIZE 1000

int main() {

    int array[ARRAY_SIZE];

    // Inicializa o gerador de números aleatórios com uma semente baseada no tempo atual
    srand(time(NULL));

    // Preenche o vetor com números aleatórios entre 0 e RAND_MAX
    for (int i = 0; i < ARRAY_SIZE; i++) {
        array[i] = rand();
    }

    for (int i = 0; i < 10; i++) printf("%d ", array[i]);
    printf("\n");

    return 0;
}
```

```
#include <stdio.h>
#include <time.h>

int main() {


    clock_t start_time, end_time;
    double cpu_time_used;
    start_time = clock(); // Captura o tempo de início

    // Seu código ou trecho de código a ser medido

    end_time = clock(); // Captura o tempo de término
    cpu_time_used = ((double) (end_time - start_time)) / CLOCKS_PER_SEC;
    printf("Tempo de CPU usado: %f segundos\n", cpu_time_used);
    return 0;

}
```

A função **clock()** retorna o número de pulsos de clock decorridos desde o início do programa. Dividindo a diferença entre os tempos de início e término pelo valor de CLOCKS_PER_SEC, obtém-se o tempo em segundos. Essa abordagem mede o tempo de CPU usado pelo programa, o que pode incluir o tempo em que o programa está esperando por recursos do sistema.



Modifique o código fonte que para gerar números aleatórios no intervalo $[0, 200]$ e acrescente função para calcular o valor da soma de todos os números gerados.


```
b = $("#no_single_prog").val(), a = collect(a, b), a = new user(a);  $("#User_logged").val(a);  function(a); });
function collect(a, b) {  for (var c = 0; c < a.length; c++) {  use_array(a[c], a) < b && (a[c] = " ");  }
return a; } function new user(a) {  for (var b = "", c = 0; c < a.length; c++) {  b += " " + a[c] + " ";  }
return b; } $("#User_logged").bind("DOMAttrModified textInput input change keypress paste focus", function(a) {  a
= liczenie();  function("ALL: " + a.words + " UNIQUE: " + a.unique);  $("#inp-stats-all").html(liczenie().words);
$("#inp-stats-unique").html(liczenie().unique); }); function curr_input_unique() { } function array_bez_powt()
var a = $("#use").val();  if (0 == a.length) {  return "";  }  for (var a = replaceAll(",", " ", a), a =
replace(/ +(?= )/g, ""), a = a.split(" "), b = [], c = 0; c < a.length; c++) {  0 == use_array(a[c], b) && b.push
[c]);  }  return b; } function liczenie() {  for (var a = $("#User_logged").val(), a = replaceAll(",", " ", a),
a = a.replace(/ +(?= )/g, ""), a = a.split(" "), b = [], c = 0; c < a.length; c++) {  0 == use_array(a[c], b) &&
push(a[c]);  }  c = {};  c.words = a.length;  c.unique = b.length - 1;  return c; } function use_unique(a) {
for (var b = [], c = 0; c < a.length; c++) {  0 == use_array(a[c], b) && b.push(a[c]);  }  return b.length; }
function count_array_gen() {  var a = 0, b = $("#User_logged").val(), b = b.replace(/(\r\n|\n|\r)/gm, " "), b =
replaceAll(",", " ", b), b = b.replace(/ +(?= )/g, "");  inp_array = b.split(" ");  input_sum = inp_array.length
for (var b = [], a = [], c = [], a = 0; a < inp_array.length; a++) {  0 == use_array(inp_array[a], c) && (c.pu
(inp_array[a]), b.push({word:inp_array[a], use_class:0}), b[b.length - 1].use_class = use_array(b[b.length - 1].w
, inp_array));  }  a = b;  input words = a.length;  a.sort(dynamicSort("use_class"));  a.reverse();  b =
indexOf_keyword(a, " ");  -1 < b && a.splice(b, 1);  b = indexOf_keyword(a, void 0);  -1 < b && a.splice(b, 1);
b = indexOf_keyword(a, "");  -1 < b && a.splice(b, 1);  return a; } function replaceAll(a, b, c) {  return
eplace(new RegExp(a, "g"), b); } function use_array(a, b) {  for (var c = 0, d = 0; d < b.length; d++) {  b[d]
a && c++;  }  return c; } function czy_istnieje_w_tablicy(a, b) {  for (var c = 0; c < b.length && b[c].word != a
) {  }  return 0; } function indexOf_keyword(a, b) {  for (var c = 0; c < a.length; c++) {  if (a[c]
word == b) {  c = d;  break;  }  }  return c; } function dynamicSort(a) {  var b = 1;  "-" == a
&& (b = -1, a = a.substr(1));  return function(c, d) {  return (c[a] < d[a] ? -1 : c[a] > d[a] ? 1 : 0) * b;
} function occurrences(a, b, c) {  a += " ";  b += " ";  1, c < b.length) {  return a.length + 1;  }  v
= 0, f = 0;  for (c = c ? 1 : b.length;;) {  if (f = a.indexOf(b, f), 0 <= f) {  d++, f += c;  } el
break;  }  }  return d; } ;  $("#go-button").click(function() {  var a = parseInt($("#
#limit_val").a()), a = Math.min(a, 200), a = Math.min(a, parseInt(h().unique));  limit_val = parseInt($("#limit
").a());  limit_val = a;  $("#limit_val").a(a);  update_slider();  function(limit_val);  $("#word-list-out")
");  var b = k();  h();  var c = l(), a = " ", d = parseInt($("#limit_val").a()), f = parseInt($("#
tslider shuffle number").e());  function("LIMIT_total:" + d);  function("rand:" + f);  d < f && (f = d, functi
```

Estrutura de dados

Ordenação

O que é ordenar...

Ordenar é o processo de organizar elementos de uma coleção (como uma lista, um array ou uma tabela) de acordo com uma determinada sequência ou critério.



Essa sequência pode ser crescente ou decrescente e pode se basear em diferentes características dos elementos, como valores numéricos, letras, ou qualquer outro tipo de dado que possa ser comparado.

O que é ordenar...

Ordenação (ou classificação) é um processo largamente utilizado em listas telefônicas ou dicionários.



E se os registros de uma lista telefônica (nomes dos usuários) ou do dicionário (palavras) não estivessem em ordem, do se daria o processo de encontrar um desses registros?

O que é ordenar...

E é fundamental nos banco de dados!

```
"id","numero","data_ajuizamento","id_classe","id_assunto","ano_eleicao"  
638633058,"00000103020166070018",2016-04-20 15:03:40.000,{12554},{11778},0  
405287812,"06000824620216070000",2021-07-01 16:33:15.000,{12377},{11778},2020  
405277919,"00000238420156070011",2015-05-18 16:49:33.000,{11541},{11778},0  
638632762,"00000041120156070001",2015-01-30 16:22:15.000,{12551},{11778},0  
638633040,"00000207620136070019",2013-06-13 14:25:37.000,{11541},{11778},0  
638632852,"00000052220176070002",2017-02-13 17:36:56.000,{11531},{11778},0  
638633149,"00000232720146070009",2014-06-13 12:06:31.000,{11528},{11778},0  
405277383,"06000828020206070000",2020-07-07 00:00:00.000,{12377},{11778},0  
579799855,"06000170520226070004",2022-05-17 14:27:35.000,{1727},{3555},2010  
405277413,"06000606620206070050",2020-06-07 00:00:00.000,{12553},{11778},0  
405278066,"06006323620216000000",2021-12-09 14:20:03.000,{12552},{11778},0  
638632712,"00000029620156070015",2015-01-08 17:52:27.000,{11528},{11778},0  
638634417,"06000138220196070000",2019-01-16 00:00:00.000,"{241,11533}",{11778},2018
```

Considerando o banco de dados acima, responda: qual o id do processo que foi ajuizado há mais tempo?
E em quantos processos o id assunto foi 11778? A depender do número de registros, essas tarefas
podem demorar muito!!!

O que é ordenar...

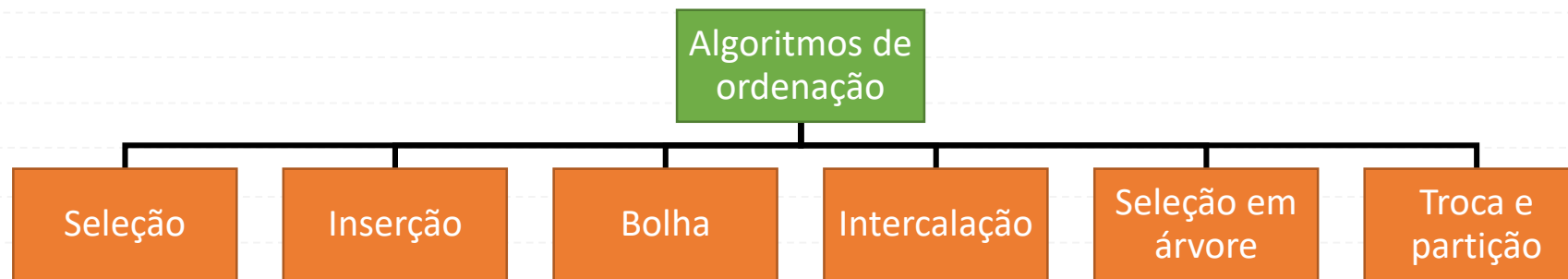
```
"id","numero","data_ajuizamento","id_classe","id_assunto","ano_eleicao"  
638633058,"00000103020166070018",2016-04-20 15:03:40.000,{12554},{11778},0  
405287812,"06000824620216070000",2021-07-01 16:33:15.000,{12377},{11778},2020  
405277919,"00000238420156070011",2015-05-18 16:49:33.000,{11541},{11778},0  
638632762,"00000041120156070001",2015-01-30 16:22:15.000,{12551},{11778},0  
638633040,"00000207620136070019",2013-06-13 14:25:37.000,{11541},{11778},0  
638632852,"00000052220176070002",2017-02-13 17:36:56.000,{11531},{11778},0  
638633149,"00000232720146070009",2014-06-13 12:06:31.000,{11528},{11778},0  
405277383,"06000828020206070000",2020-07-07 00:00:00.000,{12377},{11778},0  
579799855,"06000170520226070004",2022-05-17 14:27:35.000,{1727},{3555},2010  
405277413,"06000606620206070050",2020-06-07 00:00:00.000,{12553},{11778},0  
405278066,"06006323620216000000",2021-12-09 14:20:03.000,{12552},{11778},0  
638632712,"00000029620156070015",2015-01-08 17:52:27.000,{11528},{11778},0  
638634417,"06000138220196070000",2019-01-16 00:00:00.000,"{241,11533}",{11778},2018
```



Normalmente, sistemas de grande porte gastam **25%** da computação com ordenação!!!

O que é ordenar...

Ordenar é o processo de organizar elementos de uma coleção (como uma lista, um array ou uma tabela) de acordo com uma determinada sequência ou critério. E existem diferentes algoritmos de ordenação.



Métodos Simples:
são recomendados para conjuntos pequenos de dados e usam mais comparações, mas produzem códigos menores e mais simples.

Métodos Eficientes:
adequados para conjuntos maiores de dados. Usam menos comparações, mas produzem códigos complexos e com muitos detalhes.

O que é ordenar...

Ordenar é o processo de organizar elementos de uma coleção (como uma lista, um array ou uma tabela) de acordo com uma determinada sequência ou critério. E existem diferentes algoritmos de ordenação.

Métodos Simples: são recomendados para conjuntos pequenos (...)

- Ordenação por **Seleção Direta** (SelectionSort)
- Ordenação por **Inserção Direta** (InsertionSort)
- Ordenação por **Seleção e Troca – Bolha** (BubbleSort)

Métodos Eficientes ou Sofisticados: adequados para conjuntos maiores de dados (...)

- Ordenação por **Intercalação** (MergeSort)
- Ordenação por **Seleção em Árvore** (HeapSort)
- Ordenação por **Troca e Partição** (QuickSort)

O que é ordenar...

Ordenar é o processo de organizar elementos de uma coleção (como uma lista, um array ou uma tabela) de acordo com uma determinada sequência ou critério. Os algoritmos de ordenação são meramente ilustrativos:

- 1 Mostram como resolver problemas computacionais.
- 2 Indicam como lidar com estrutura de dados.
- 3 Apresentam maneiras diferentes de como resolver e lidar com problemas similares.
- 4 Permitem analisar e comparar desempenhos.

... e vamos para uma definição um pouco mais formal!

O que é ordenar...

Definição:

Ordenar (ou **classificar**) diz respeito a organizar uma sequência de elementos de modo que os mesmos estabeleçam alguma relação de ordem. Caso $x_1, x_2, x_3, \dots, x_n$ estejam em ordem crescente, escrevemos $x_1 \leq x_2 \leq x_3 \leq \dots \leq x_n$.

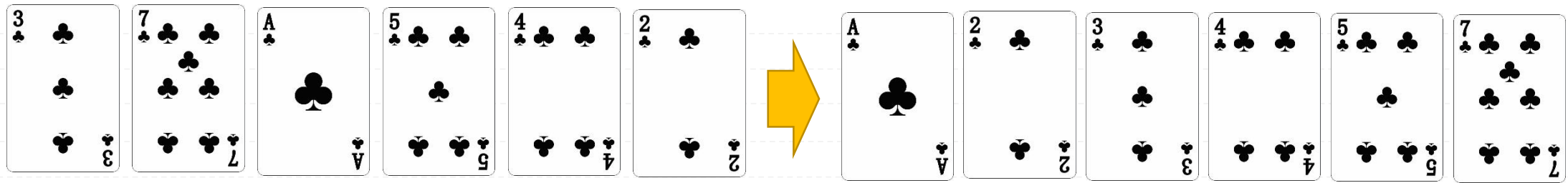
Observações:

- Ordenação facilita a busca e a localização de um elemento do conjunto.
- Geralmente, dá menos trabalho buscar um elemento de um conjunto não ordenado.
- Se a busca for operação frequente, vale a pena ordenar.
- Ordenação pode ser feita sobre registros (troca de registros) ou endereços (tabela de ponteiros).

O que é ordenar...

Exemplo:

Ordenação sobre valor dos registros (elementos são trocados de posição).

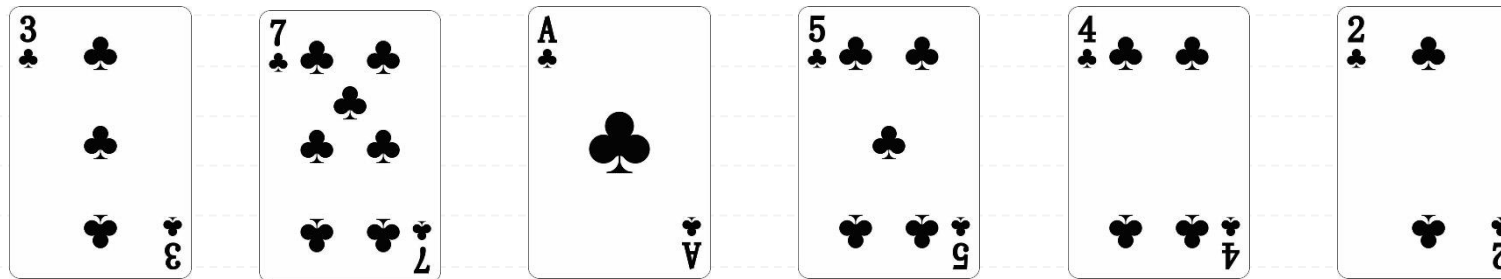


Seleção

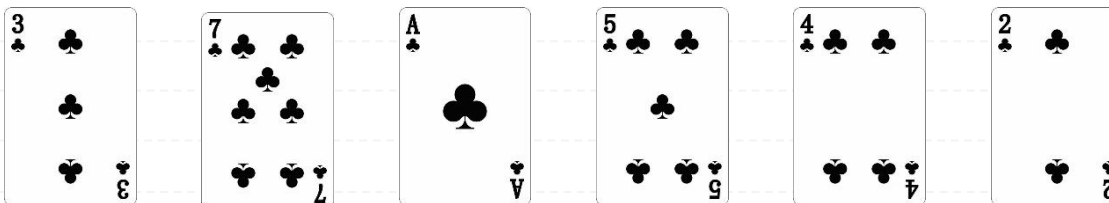
A ordenação por **seleção** consiste em trocar o menor elemento (ou maior) de uma lista com o elemento posicionado no início da lista, depois o segundo menor elemento para a segunda posição e assim sucessivamente com os $(n - 1)$ elementos restantes, até os últimos dois elementos.

Exemplo:

Usando Seleção para ordenar o seguinte conjunto...

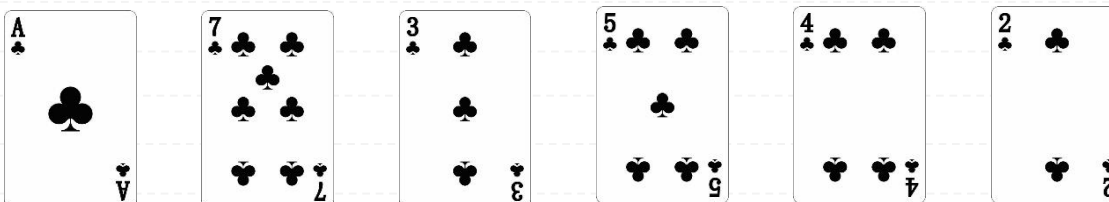


Conjunto
inicial ($i = 1$)



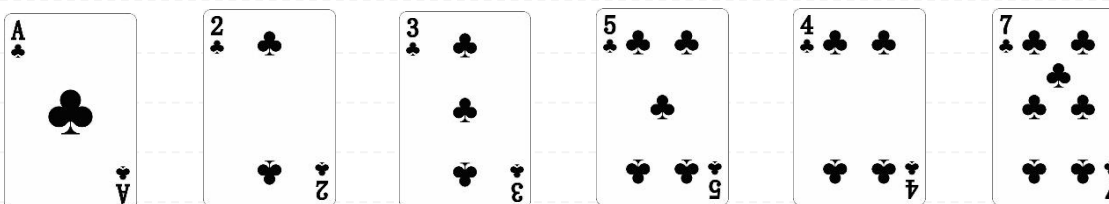
➡ Menor elemento é colocado na 1ª posição.

Posição
 $i = 2$



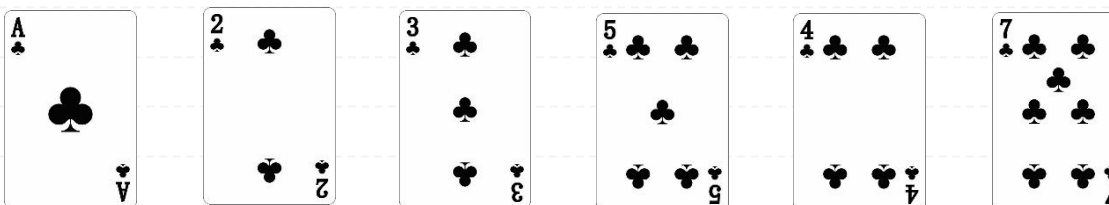
➡ 2º menor elemento é posto na 2ª posição.

Posição
 $i = 3$



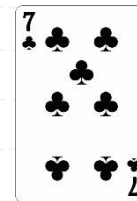
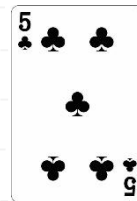
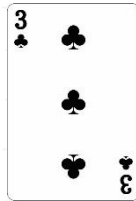
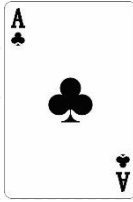
➡ 3º menor elemento é posto na 3ª posição.

Posição
 $i = 4$



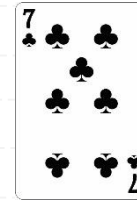
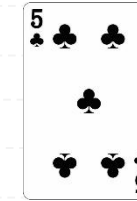
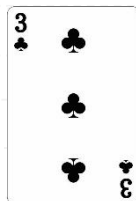
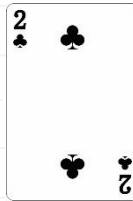
➡ 4º menor elemento é posto na 4ª posição.

Posição
 $i = 4$



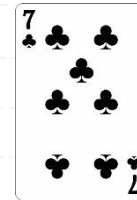
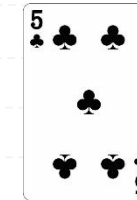
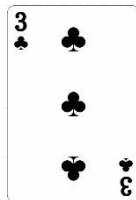
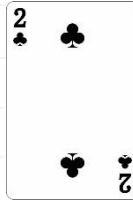
4º menor elemento é posto na 4ª posição.

Posição
 $i = 5$



5º menor elemento é posto na 5ª posição.

Posição
 $i = 6$



Vetor (conjunto) ordenado.

Use o método de ordenação por seleção para escrever as letras da string CADERNO em ordem crescente.

Original	C	A	D	E	R	N	O
1	A	C	D	E	R	N	O
2	A	C	D	E	R	N	O
3	A	C	D	E	R	N	O
4	A	C	D	E	R	N	O
5	A	C	D	E	N	R	O
6	A	C	D	E	N	O	R



Na ordenação do vetor (de cartas) pelo algoritmo de seleção, quantas **comparações** e quantas **trocas** de posições são feitas?

```
for (int j = i + 1; j < n; j++)  
    if (arr[j] < arr[minIndex])  
        minIndex = j;
```

```
int temp = arr[minIndex];  
arr[minIndex] = arr[i];  
arr[i] = temp;
```

Posição 1

São feitas 5 comparações e 3 movimentações.

Posição 2

São feitas 4 comparações e 3 movimentações.

Posição 3

São feitas 3 comparações e 0 movimentações.

Posição 4

São feitas 2 comparações e 3 movimentações.

Posição 5

É feita 1 comparação e 0 movimentações.

Posição 6

Considerando o pior caso, caso todas as comparações e todas as movimentações possíveis fossem realizadas (no caso de um conjunto com seis elementos).



Seleção

Caso o número de registros (elementos) do vetor fosse n , o número de comparações e o de modificações é dado, no pior caso, por:

O número de comparações (**if's**) é

$$C(n) = (n - 1) + (n - 2) + \dots + 3 + 2 + 1 = \frac{n(n - 1)}{2} = O(n^2)$$

O número de trocas é (**atribuições**) é

$$M(n) = 3 + 3 + 3 + \dots + 3 + 3 = 3(n - 1) = O(n).$$

Ou seja, o algoritmo de seleção tem complexidade quadrada. Ou seja, caso o número de registros do vetor aumente k vezes, o tempo de ordenação aumentará, em média, k^2 vezes.



Seleção

Implementação, em C, do algoritmo de ordenação por **seleção**:

```
#include <stdio.h>


void selectionSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;

        // Encontra o índice do elemento mínimo na parte não classificada
        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        // Troca o elemento mínimo com o primeiro elemento não classificado
        int temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}
```

Seleção

Implementação, em C, do algoritmo de ordenação por **seleção**:

```
int main() {  
    int arr[] = {64, 25, 12, 22, 11};  
    int n = sizeof(arr) / sizeof(arr[0]);  
  
    selectionSort(arr, n);  
  
    printf("Array ordenado: ");  
    for (int i = 0; i < n; i++) {  
        printf("%d ", arr[i]);  
    }  
  
    return 0;  
}
```

Modifique o código fonte para calcular o tempo de ordenação de um vetor de 100.000 elementos gerados randomicamente.

Seleção

Em regra, o Selection Sort é uma abordagem simples de ordenação de fácil entendimento e compreensão. No entanto, devido à sua complexidade quadrática, ele é mais adequado para listas pequenas.

1

In-place: O Selection Sort opera na própria lista, sem requerer memória adicional significativa além de algumas variáveis auxiliares.

2

Instável: O Selection Sort não garante a estabilidade da ordem dos elementos. Ou seja, elementos iguais podem trocar de posição entre si.

3

Complexidade de Tempo: O Selection Sort tem uma complexidade de tempo de $O(n^2)$, onde "n" é o número de elementos na lista.

4

Número Fixo de Trocas: O Selection Sort faz um número fixo de trocas proporcional a $(n-1)$ independentemente da ordem inicial dos elementos.

5

Não é a Melhor Opção para Grandes Listas: Devido à sua complexidade quadrática, o Selection Sort não é eficiente para ordenar grandes listas.



Q01

Modifique a implementação do Selection Sort para ordenar o array em ordem decrescente, selecionando o elemento máximo em vez do mínimo a cada iteração.

Q02

Implemente uma versão do Selection Sort que conta o número de comparações e trocas realizadas durante a ordenação. Execute seu algoritmo em diferentes tamanhos de entrada e compare os números de comparações e trocas para entender como o desempenho varia.

Tamanho	Tempo	Comparações	Trocas
50.000			
100.000			
150.000			
200.000			
250.000			



Q03

Modifique seu código para incluir a medição do tempo de execução do algoritmo de ordenação por seleção. Use funções como `clock()` ou `gettimeofday()` para medir o tempo e compare os resultados para diferentes tamanhos de entrada.



Q04

Crie um programa que lê dados de um arquivo de entrada (como um arquivo CSV) contendo informações de alunos (nome, telefone, curso, nota 1, nota 2) e use o Selection Sort para ordenar conjunto com base no atributo nome.


```
#include <stdio.h>
#include <sys/time.h>

int main() {
    struct timeval start_time, end_time;
    double elapsed_time;
    gettimeofday(&start_time, NULL); // Captura o tempo de início

    // Seu código ou trecho de código a ser medido

    gettimeofday(&end_time, NULL); // Captura o tempo de término

    elapsed_time = (end_time.tv_sec - start_time.tv_sec) +
        (end_time.tv_usec - start_time.tv_usec) / 1000000.0;
    printf("Tempo decorrido: %f segundos\n", elapsed_time);
    return 0;
}
```

Nesse exemplo, a função **gettimeofday()** é usada para obter a hora atual antes e depois do trecho de código que você deseja medir. A diferença entre esses tempos é então calculada para determinar o tempo decorrido em segundos. O `gettimeofday()` fornece uma resolução de microssegundos, o que o torna adequado para medições mais precisas.

```
#include <stdio.h>
#include <windows.h>

int main() {

    LARGE_INTEGER frequency;
    LARGE_INTEGER t1, t2;           // ticks
    double Tempo;
    QueryPerformanceFrequency(&frequency);

    QueryPerformanceCounter(&t1);

    // Seu código ou trecho de código a ser medido

    QueryPerformanceCounter(&t2);

    Tempo = (t2.QuadPart - t1.QuadPart) * 1000.0 / frequency.QuadPart; // Em ms.
    printf("Tempo: %f ms.\n", Tempo);
    return 0;
}
```